Más funciones sobre listas

Taller de Álgebra I

Segundo cuatrimestre de 2015

Ejercicios sobre Listas

Implementar las siguientes funciones

- `pertenece :: Integer -> [Integer] -> Bool`
  que indica si un elemento aparece en la lista.
- Implementar la función `hayRepetidos :: [Integer] -> Bool`
  que indica si una lista tiene elementos repetidos.
- `menores :: Integer -> [Integer] -> [Integer]`
  que calcula los elementos de la lista que son menores al primer parámetro.

Últimas

- `quitar :: Integer -> [Integer] -> [Integer]`
  elimina la primera aparición del elemento en la lista (de haberla).
- `maximo :: [Integer] -> Integer`
  que dada una lista no vacía calcula el mayor elemento de la misma.

Cambio de base en Haskell

- **Problema:** Dados $a, b \in \mathbb{Z}_+$ con $b > 1$, escribir el número a en base b.
- Por ejemplo ...
  1. Si $a = 10$ y $b = 2$, entonces la respuesta es 1010.
  2. Si $a = 17$ y $b = 3$, entonces la respuesta es 122.
  3. Si $a = 145$ y $b = 10$, entonces la respuesta es 145.

Implementar en Haskell

- La función `enBase :: Integer -> Integer -> [Integer]`
  Esta función toma un número y lo transforma a la base pasada como segundo parámetro. Para ello devuelve una lista representando al número. Ejemplo:
  `enBase 3 17` $\rightsquigarrow$ `[1,2,2]`.
- La función `deBase :: Integer -> [Integer] -> Integer`
  que toma una base y un número (en forma de la lista de sus dígitos) y devuelve el número como `Integer`.
  `deBase 3 [1,2,2]` $\rightsquigarrow$ 17.

Ejercicios Parte 1: La conjetura del 196

Los números de Lychrel (`http://gaussianos.com/la-conjetura-del-196/`)

- Tomemos un número natural cualquiera, por ejemplo el 180
- Si damos vuelta sus cifras (081) y lo sumamos al original nos queda $180 + 081 = 261$
- Ahora repetimos el proceso con el 261: $261 + 162 = 423$
- Lo mismo con el 423: $423 + 324 = 747$. ¡Un número capicúa!
- ¿Será verdad que para cualquier natural, siempre llegamos a un número capicúa?

Los llamados números de Lychrel son los números naturales en base 10 que nunca llegan a dar un número capicúa como resultado del proceso.

Implementar

- La función `capicuaPara :: [Integer] -> [Integer]` que toma un número (es decir sus dígitos en base 10) y realiza el proceso antedicho hasta que se consiga su número capicúa asociado.
  Ejemplo: `capicuaPara [1,8,0]` $\rightsquigarrow$ `[7,4,7]`.
- Preparar el "ctrl+c" y ejecutar capicuaPara [1,9,6], ¿qué ocurre?

Ejercicios Parte 2

1. Escribir una función que cambie de base un número. Debe tomar dos enteros `b1` y `b2` y un entero `a` representado en base `b1` (expresado como una lista de enteros) y retornar `a` en base `b2` (también expresado como una lista de enteros).
2. Escribir una función que indique si una lista de números enteros está ordenada de manera no-decreciente.
3. Escribir una función que elimine los números repetidos, de haberlos, dejando sólo la última aparición de cada uno.
4. Idem anterior, pero dejando la primera aparición de cada uno.
5. Escribir una función que reciba dos listas ordenadas en forma no-decreciente, y que retorne una lista que contenga la unión de los elementos de las dos listas recibidas, y que también esté ordenada en forma no-decreciente. Por ejemplo, al pasarle $[2, 3, 7]$ y $[1, 3, 3, 9]$, debería devolver $[1, 2, 3, 3, 3, 7, 9]$.
6. (difícil) Escribir una función que dada una lista, retorne la misma pero ordenada de menor a mayor.